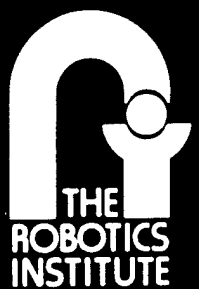
Resolution Independent Grid-based Path Planning

Gita Krishnaswamy and Anthony Stentz

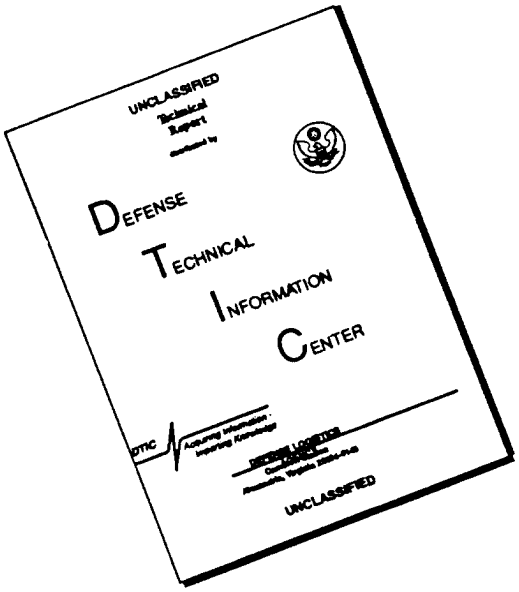
CMU-RI-TR-95-08



Carnegie Mellon University
The Robotics Institute

Technical Report

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Resolution Independent Grid-based Path Planning

Gita Krishnaswamy and Anthony Stentz

CMU-RI-TR-95-08

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

April 1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1995	technical

4. TITLE AND SUBTITLE
Resolution Independent Grid-based Path Planning

5. FUNDING NUMBERS

6. AUTHOR(S)
Gita Krishnaswamy and Anthony Stentz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU-RI-TR-95-08

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Energy conservation in a rover is an important factor which should be considered for a mission of long duration. Locomotion is one of the primary consumers of energy. The energy used depends on the terrain and the path taken by the robot. This paper develops a planning strategy based on cell decomposition and A* algorithm which would minimize power usage due to locomotion. Cell decomposition is used because of its ability to represent the environment as a grid of continuous values. The current limitation with cell decomposition is that the path produced is resolution-optimal only. The method developed in this paper overcomes this problem and produces resolution-independent optimal solutions for a binary (obstacle/free space) environment and better results for the continuously varying environment than common existing techniques. This is done in a computationally efficient manner.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	26 pp
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unlimited	unlimited	unlimited	unlimited

Contents

List of Figures

Abstract

Energy conservation in a rover is an important factor which should be considered for a mission of long duration. Locomotion is one of the primary consumers of energy. The energy used depends on the terrain and the path taken by the robot. This paper develops a planning strategy based on cell decomposition and A* algorithm which would minimize power usage due to locomotion. Cell decomposition is used because of its ability to represent the environment as a grid of continuous values. The current limitation with cell decomposition is that the path produced is resolution-optimal only. The method developed in this paper overcomes this problem and produces resolution-independent optimal solutions for a binary (obstacle/free space) environment and better results for the continuously varying environment than common existing techniques. This is done in a computationally efficient manner.

1 Introduction

Autonomous mobile robots are useful for a number of applications ranging from factory automation to planetary exploration. This work is specifically related to the lunar rover initiative which is to robotically roam the moon for hundreds of kilometers, fulfilling the dual agendas of entertainment and scientific exploration.

An autonomous robot fails if it enters a situation from which it cannot recover. This factor and the long duration of the mission requires the robot to be highly reliable in its entire system, from mechanical configuration to the planning strategy. One of the most important factors constraining the capability of a robot is power, which is required for locomotion, communication, sensing and computing. Therefore, the conservation of power is very important and the planning strategy used for the lunar rover should be optimal in energy consumption.

Since energy consumption during locomotion is a large part of the total power requirement, determining a path that minimizes energy during traversal becomes an important task that the robot must perform. It is also necessary for the robot to compute this path quickly in real time.

2 Problem Statement

The problem of finding optimal paths for autonomous mobile robot through environments with obstacles has attracted much research interest. Path planning is usually based on a few general approaches that can be broadly classified into two groups - network/graph models and grid-based models.

Network/graph models are dependent on the nature and clutter of the environment. These models represent the environment as a network of free-space regions or as a graph of obstacle vertices. An example of this approach is the roadmap or visibility graph method. The roadmap or visibility graph approach consists of capturing the connectivity of the robot's free space in a network and then searching for the shortest path in this network

that connects the start and goal points. This is illustrated in Fig.1a where the true optimal path is found. The disadvantages of this method are that only a binary representation of the world (obstacle/free space) is possible and the order of complexity is a function of the number of obstacles. Also, accurate sensor information is needed in order to represent the free space region.

Grid-based models, such as the cell decomposition approach, impose structure on the environment without regard to the nature or clutter of the environment. The cell decomposition approach consists of decomposing the world into simple regions called cells. A connectivity tree is then constructed representing the adjacency relation between the cells and searched for the shortest path or sequence of cells that connects the start and goal points. Fig.1b illustrates this technique for a binary representation where the path is determined by making transitions to the 8 adjacent cells. The disadvantage with this method is that the path found is resolution-optimal only, i.e., the best path is constrained to follow the 8-connected cell transitions, and therefore is not truly optimal. In many cases, as the cells decrease in size and connectivity of the cells is increased, the true optimal path is approached. However the complexity of the problem (which is a function of the number of cells) increases rapidly and determining the path becomes very time consuming. Fig.1c shows that when the resolution is doubled, the path determined is closer to the true optimal but the search space also doubles. The advantage of this method over the visibility graph method is that the world can be represented in a continuous manner and for the binary case, the number of obstacles present is irrelevant.

Another grid-based model is the potential field approach which consists of representing the space as "attractive potential" (like the goal point) and "repulsive potential" (such as obstacles) and moving the robot by computing the gradient of the potential in its neighborhood. Its disadvantage is that it can get trapped in a local minima.

The lunar surface is a natural, rough terrain. Therefore binary representation (obstacle/ nonobstacle) of the terrain is insufficient and continuous cost values are needed to represent it. Given the major planning paradigms discussed above, for this problem, only cell decomposition makes sense for continuous cost values. The problem with the cell decomposition

approach, however is that they are resolution-optimal only as illustrated above. Ways to get around the resolution-optimal problem in the cell decomposition approach and get a truly optimal solution are being explored.

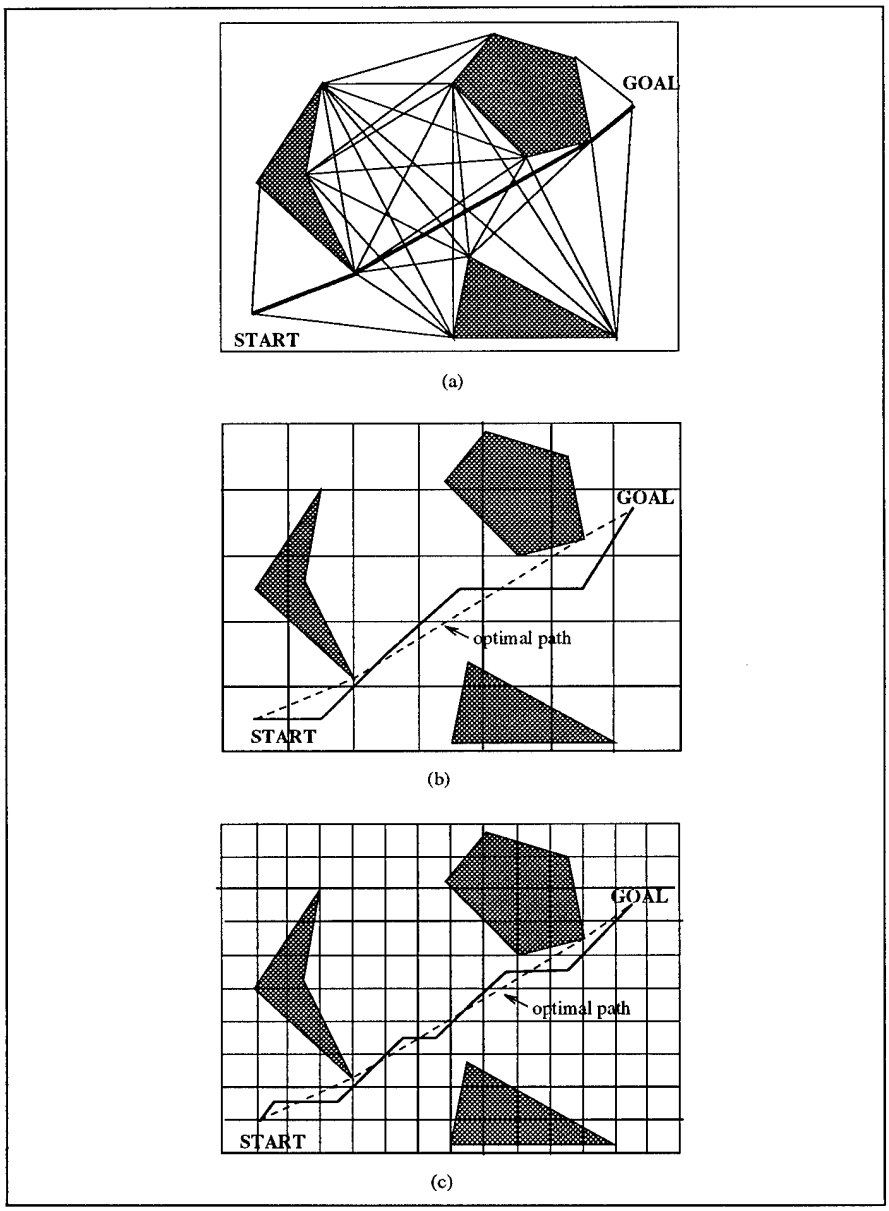


Figure 1: (a) Path determined by visibility graph method, (b) Path determined by cell decomposition with 8-connectivity, (c) Path determined by cell decomposition with increased resolution

3 Prior Work

Various aspects of path planning for a mobile robot like representation of the environment, search techniques and complexity of search have been addressed in the literature.

Lozano-Perez [Lozano-Perez 1983] presented an approach in "spatial planning" based on characterizing the robot as a single point in configuration space, characterizing the regions forbidden to the robot as configuration space obstacles and then determining the exact path.

Khatib [Khatib 1986] pioneered the potential field approach. Barraquand and Latombe [Barraquand and Latombe 1991] combined this approach with other random techniques to escape from local minima. This was achieved by tracking the valleys of the potential function and using numerical tracking techniques to find the global minimum.

Taylor [Taylor 1976] investigated the problem of motion planning in the presence of uncertainty in which he used numerical uncertainty propagation techniques. This method was improved by using symbolic propagation techniques [Brooks 1982] instead of numerical ones. Dufay and Latombe [Dufay and Latombe 1984] solved this problem based on inductive learning.

Some search strategies were explored by Jarvis [Jarvis 1985] and Nilsson [Nilsson 1980]. Jarvis used distance transforms to generate path planning solutions, which used the concept of finding the shortest path via the steepest descent. Nilsson used heuristics to Prune the search tree and therefore reduce the search space.

Most of the work in the literature assumes that the environment is completely known. Chatila [Chatila 1982], Zelinsky [Zelinsky 1992] and Stentz [Stentz 1994] have explored the problem of motion planning in partially known or unknown environments. Chatila based his planner on an exact decomposition of the empty subset of the workspace into convex cells and updating the decomposition to account for new information. Zelinsky used quadtree to model the environment (as free and obstacle space) and used distance transform to generate the path. Unknown regions were initially treated as free space and the path was refined as the robot acquired more knowledge about the environment. The use

of quadtree reduced the number of states to search and hence increased efficiency. Stentz used cell decomposition to represent a continuously varying environment and introduced a new algorithm D* (based on A*) to search the space and plan paths for the changing environment in an efficient, optimal and complete manner.

This work attempts to determine an algorithm that overcomes the limitation of the cell decomposition technique for a deterministic environment.

4 Methods

4.1 Terrain definition

The objective of the path planner is to move the robot, which is treated as a point, from some location to a goal location using a terrain model, such that the cost of traversal is minimized. In this case, the cost of traversal reflects the energy required that is strictly dependent on the terrain.

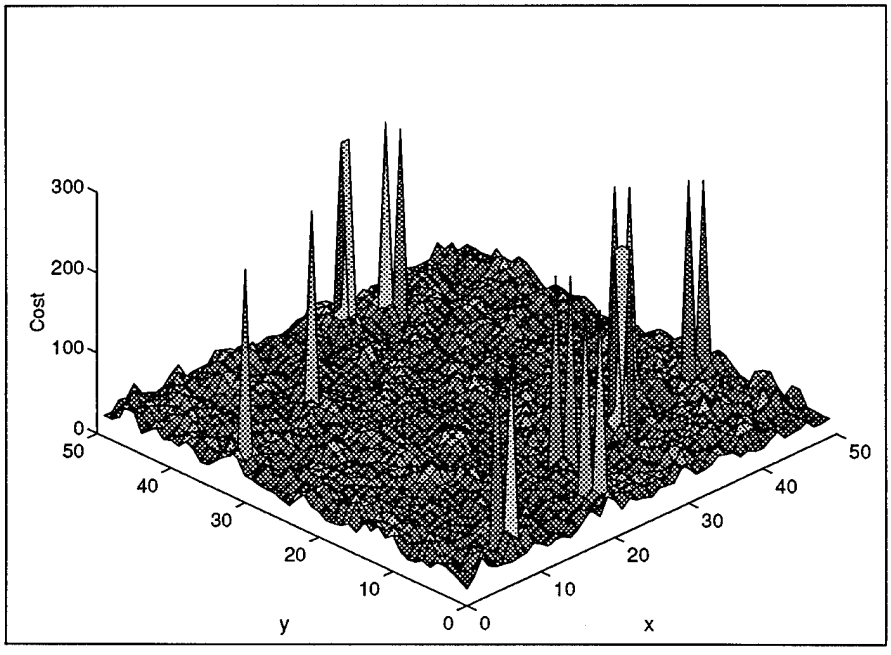


Figure 2: Randomly generated fractal terrain

The lunar terrain surface is mainly defined by meteor strikes. Due to the stochastic nature of meteor strikes, insurmountable obstacles are more random than on earth. In this project, fractal terrain generated based on a random number generator is used, which is a reasonable simulation of the moon surface. An example of such a surface is shown in Fig.2 where the z-axis reflects the cost of the terrain.

In order to perform path planning, the robot's environment or terrain model is partitioned into cells of a convenient size, with each cell having a numerical cost value (ranging from 10-255) calculated based on the terrain characteristics such as obstructions, slope and altitude. For example, steep upward slopes will require a lot of energy and such cells have high cost; craters are obstructions and are high cost. On the other hand, benign terrain and downward slopes are of low cost. Other indirect factors such as areas or cells where the antenna cannot point directly to earth and therefore requires more energy for communication are however not considered.

4.2 Path Planning definitions

The underlying algorithm used to search the decomposed space is the A*, which guarantees resolution-optimal path. The purpose of the methods to be explored is to find a way to obtain a truly optimal solution using cell decomposition and A*-based search techniques, that gives a reasonably smooth path independent of the resolution.

The problem space is formulated as a set of states denoting robot locations connected by directional arcs, each of which has the associated cost (which is the sum of distance traveled in a cell x cost of cell). The arc connects from the center of one cell to the center of another. The robot is considered to be a point. The robot starts at the START state and moves across arcs to other states until it reaches the GOAL state. This is done using an A*-based algorithm.

4.3 The A* algorithm

A* is a best-first search algorithm which uses heuristics to reduce the complexity of the search. The algorithm makes use of a heuristic evaluation function, h(X) which is an estimated cost of reaching the GOAL from state X. In addition, g(X) is the actual cost incurred in going from the START state to state X. The merit of the state or node is the total cost f(X) = h(X) + g(X). A* maintains an OPEN list of unexpanded nodes, sorted by cost, which is used to pick the next state to move to and to propagate information about changes to the arc cost function. Every state X also has an associated tag t(X), where t(X) = NEW if the node has not been on the open list, t(X) = OPEN if it is on the open list, and t(X) = CLOSED if it is no longer on the open list. Every state X except the START state has a backpointer to its previous or parent state, denoted by b(X) which represents the path.

The heuristic function h(X) must be able to provide a reasonable estimate of the merit of a node and should be inexpensive to compute. Here, h(X), the estimated cost is the Euclidean distance between X and the GOAL multiplied by the lowest cost in the grid. This ensures that it is a strict lower bound, ie., the cost is not overestimated and hence guarantees an optimal solution. A good lower bound greatly reduces the complexity because the search tree is pruned significantly.

4.4 Method 1: A* with increased look-ahead

A path determined using A* with 8-connectivity is limited to travel only in the horizontal, vertical or diagonal directions. As a result, at times there is a significant deviation between this path and the true optimal path as shown in Fig.1b. Increasing the look-ahead or search to more levels increases the choice of directions of travel (Fig.3) and hence a more straightforward (or lower cost) path could be found.

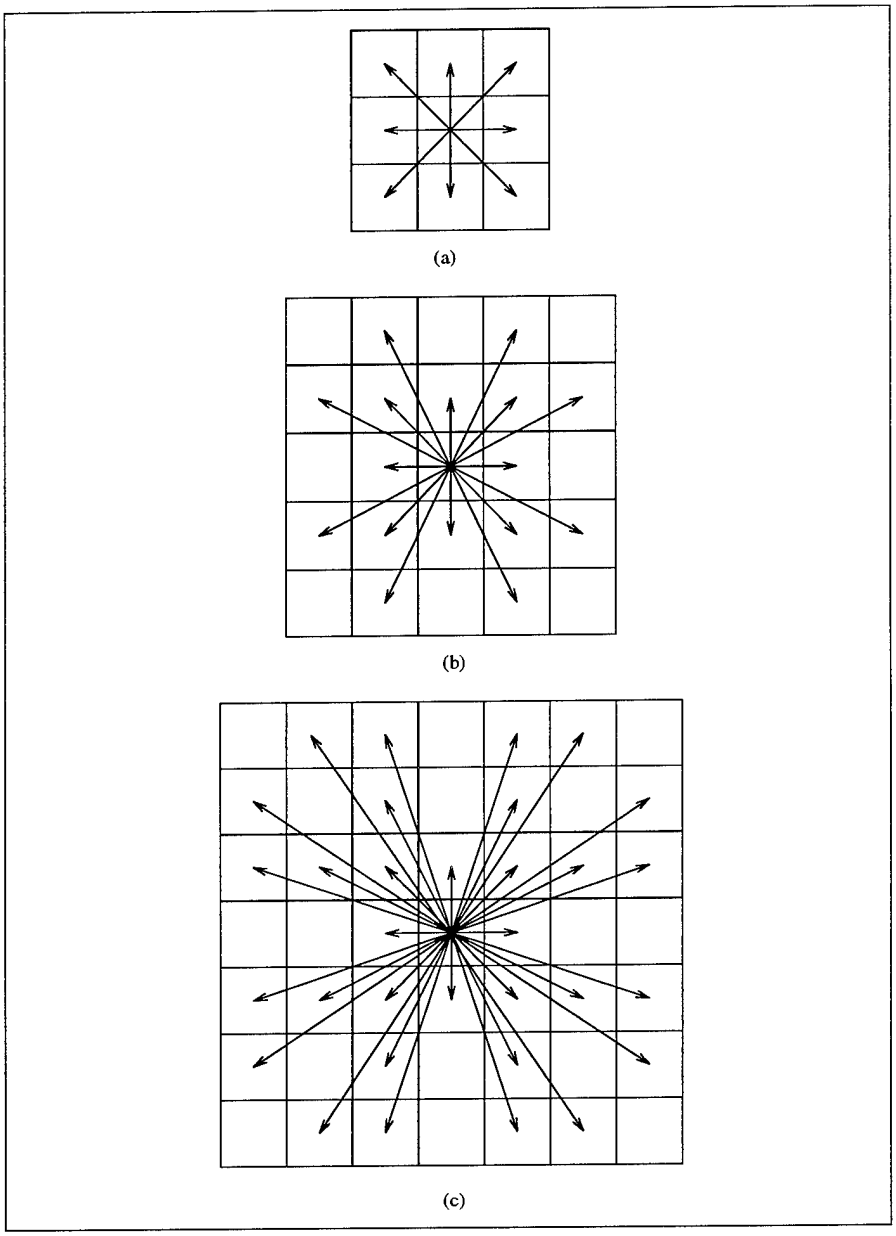


Figure 3: (a) Level1 (8-connectivity), (b) Level2 (16-connectivity) (c) Level3 (32-connectivity)

The A* algorithm with multi-levels is as follows:

```
X = MIN-STATE()
If X = NULL, Return(-1)
t(X) = CLOSED
If X = GOAL, STOP
DELETE-OPEN(X)
  For each child Y (levels 1..N) of X
    If t(Y) = NEW
      b(Y) = X
      f(Y) = g(X) + c(X,Y) +h(Y)
      INSERT-OPEN(Y)
    If ((t(Y) = OPEN) and (c(X,Y) < c(b(Y),Y)))
      b(Y)= X
      f(Y)= g(X) + c(X,Y) + h(Y)
      SORT-OPEN()
    If ((t(Y) = CLOSED) and (c(X,Y) < c(b(Y),Y)))
      b(Y)= X
      f(Y) = g(X) + c(X,Y) + h(Y)
      UPDATE-COST(Y)
CONTINUE
```

- MIN-STATE returns the state with minimum f(X).
- DELETE-OPEN(X) deletes the state X from the open list.
- t(X) gives the status of the state X- OPEN, CLOSED or NEW.
- b(X) is the state that X backpoints to.
- c(X,Y) is the actual cost to move from X to Y.
- g(X) is the actual cost to move from START to X and is c(Z,X)+g(Z) where Z is the parent of X.
- INSERT-OPEN(X) inserts the state X into the open list.

- SORT-OPEN sorts the open list such that the state with minimum f is on top.
- UPDATE-COST(X) updates the cost of all the branches originating from state X.

Optimality

The algorithm produces the optimal path for the condition that the transitions and directions of paths allowed are dictated by the resolution and the number of look-ahead levels N, and that the arcs are connected from center of one cell to center of the next. The path is however not truly optimal.

Computational time & Memory requirements

The number of children for a state X is $4 * 2^N$, where N is the number of levels used. With more levels, each state generates more children covering a wider space. Hence the computational time increases because a cell is visited many more times and time is spent in calculating different traversal costs at each visit.

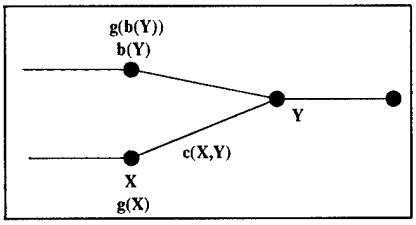
The memory required does not change significantly with increasing levels because it depends on the total number of cells and as long as that remains constant, increasing the number of levels should not increase memory required.

Dominance Criterion

Calculating the cost of traversal from a node to its child c(X,Y), where Y is the child of X, took a large part of computer time, especially at higher levels. Hence a dynamic programming dominance criterion was used to improve the speed of the algorithm as follows: If node X has a child Y and t(Y) != NEW (i.e., the parent of Y is b(Y)), in order to decide if node X should be the parent instead of b(Y),

Compute c(X,Y) only if

(g(X) + lower bound estimate of cost to move from X to Y) < g(Y).



In other words, if the cost to reach X is greater than the cost to reach Y through it's original parent b(Y), it is ensured that the path through X will be more costly.

4.5 Method 2: A* with connect-back

It was noted that with Method 1, the euclidean distance between two points in a uniform cost field could not always be determined unless a large number of levels were used, which was computationally inefficient.

Due to limitations of the previous look-ahead technique in terms of computational time, another method was devised. This technique would essentially search using A* with 4 or 8-connectivity and shorten the path if possible up to that point as the search proceeded. This is done by redefining the backpointer of a state to its parent's parent and so on as long as there is an improvement in the cost and then propagating this improvement. Fig.4a shows the path found purely by A* with 4-connectivity (horizontal/vertical moves) for a uniform cost field. Fig.4b illustrates how the basic path found by A* with 4-connectivity is shortened by the method described above. In the first figure, the path up to the point A (dashed path) is shortened in two steps to the straight line SA by connecting directly to A's parent's parent and so on. In the second figure, the path up to the point B (dashed path) is shortened to the straight line SB and finally, in the third figure, the straight line from S to G is obtained. To obtain a similar path with Method 1, a Level 4 search would have to be conducted, which means nearly 60 children have to be explored at each state. This would take significantly longer.

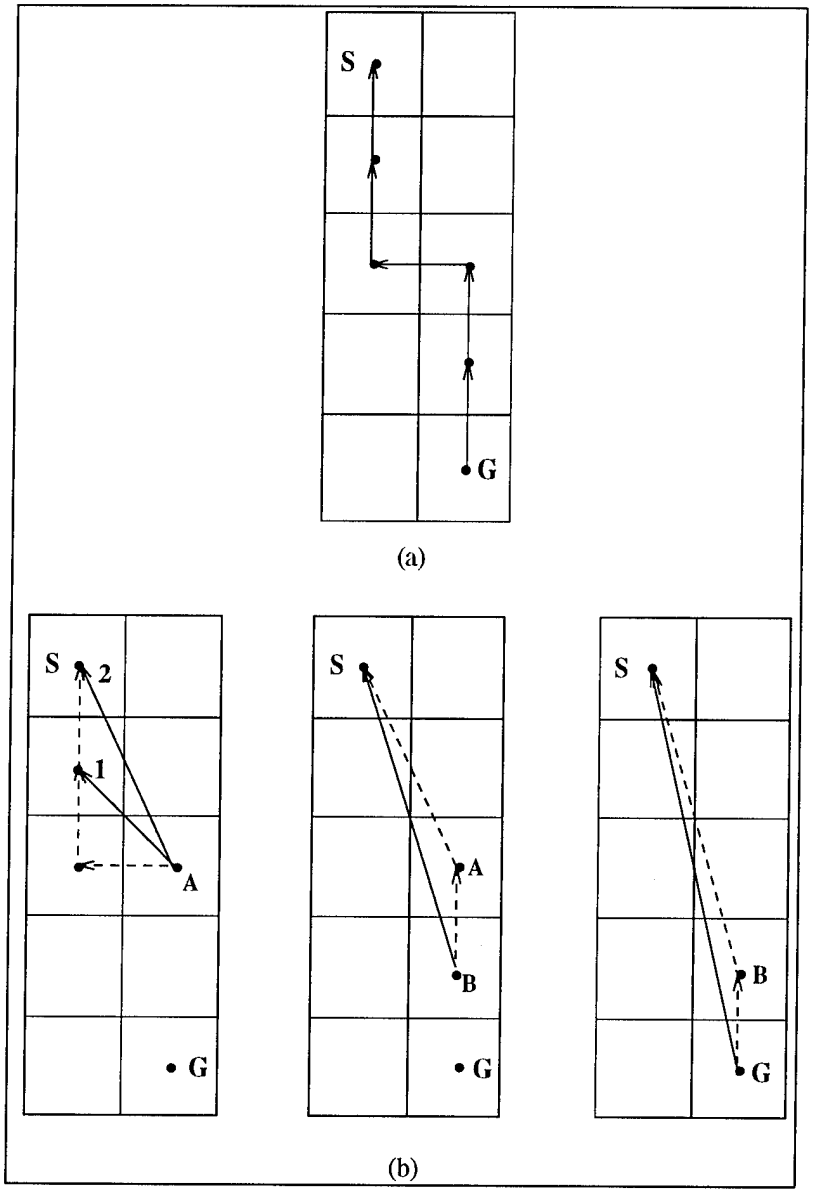


Figure 4: (a) Path found by A* with 4-connectivity (b) Path found by A* with refinement

The shortcoming with this technique however is that it does not always produce the optimal path. Trying to reduce the path at every stage results in elimination of certain nodes, and hence leads to a sub-optimal solution. This is illustrated in Fig.5 where the path found by this connect-back technique performs worse than a Level 1 search.

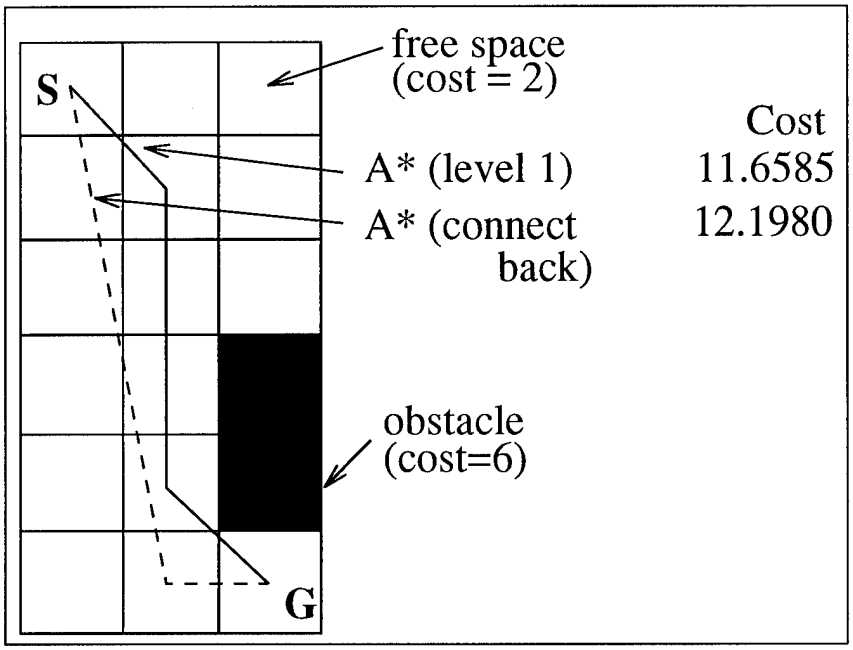


Figure 5: Paths found using Level1 search of Method 1 and the new technique of connecting back

The method was modified by adjusting the path only after a sequence of cells was found, rather than at every stage. The point where the adjustment is done is when the path snags a corner. The path is then shortened if possible from this corner node which is introduced, by redefining the backpointer of this node to its parent's parent and so on as long as there is improvement in the cost. This is shown in Fig.6b. This results in an optimal polyline up to that point. Note that the corner nodes are not involved during the search but only when the path-optimization is performed. This process is carried out until the GOAL is reached, at which point a final adjustment is made. In the first figure of Fig.6b, the corner node A is introduced when the path passes through the corner, at which point the path is shortened in two stages to the straight line SA as before. In the second figure, when the goal point G is reached, the path is reduced initially to S-A-G and further to SG as in the third figure.

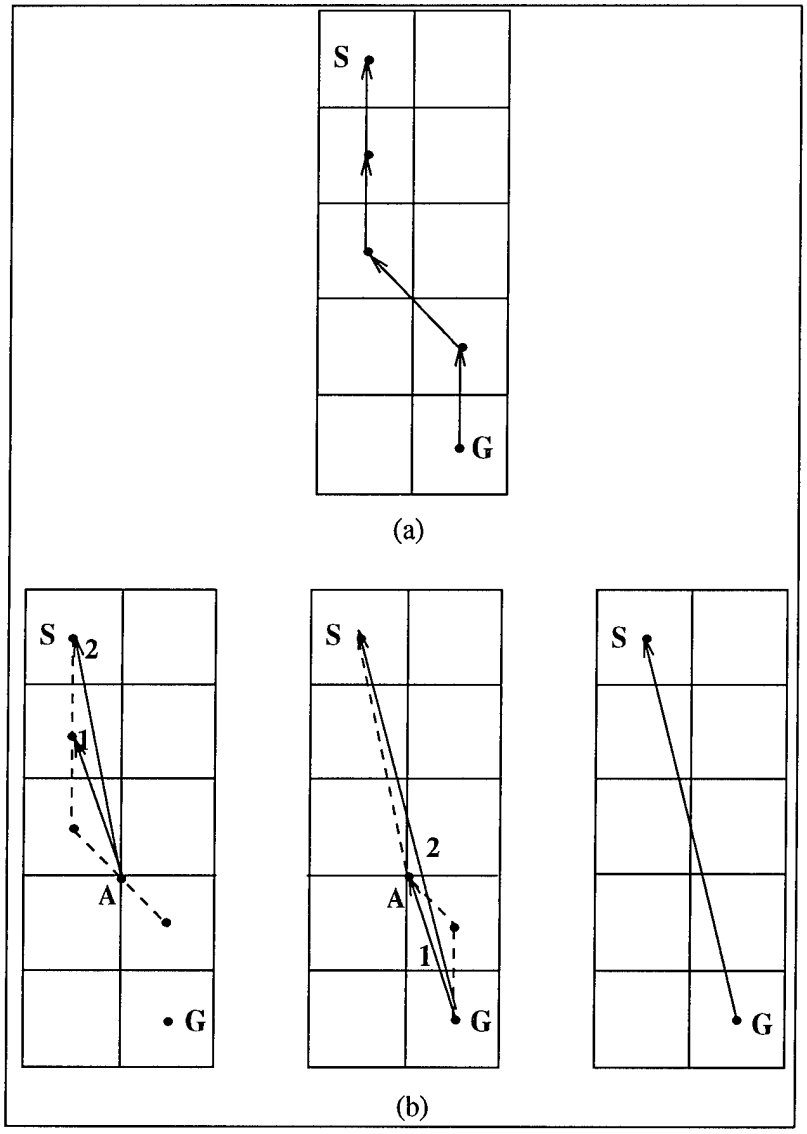


Figure 6: (a) Path found by A* with 8-connectivity (Level1) (b) Path found by A* with improved refinement, which is optimal

The algorithm is as follows:

X = MIN-STATE()

If X = NULL, Return (-1)

t(X) = CLOSED

If X = GOAL

while c(b(b(X)),X) < c(b(X),X)

b(X) = b(b(X))

f(X) = g(b(b(X)) + c(b(b(X)),X)

STOP

DELETE-OPEN(X)

If b(X) is diagonal to X

Z = CORNER-INSERT(X,b(X))

b(Z) = b(X)

b(X) = Z

while c(b(b(Z)),Z)< c(b(Z),Z)

b(Z) = b(b(Z))

f(Z) = g(b(b(Z)) + c(b(b(Z)),Z) + h(Z)

For each child Y of X

If t(Y) = NEW

b(Y) = X

f(Y) = g(X) + c(X,Y) + h(Y)

INSERT-OPEN(Y)

If ((t(Y) = OPEN) and (c(X,Y) < c(b(Y),Y)))

b(Y)= X

f(Y)= g(X) + c(X,Y) + h(Y)

SORT-OPEN()

If ((t(Y) = CLOSED) and (c(X,Y) < c(b(Y),Y)))

b(Y)= X

f(Y) = g(X) + c(X,Y) + h(Y)

UPDATE-COST(Y)

CONTINUE

- CORNER-INSERT(X,Y) creates a corner node, where X is the parent of Y and are diagonal to each other.

The dominance criterion used in Method 1 is employed here too.

Optimality

This method yields a truly optimal solution (which has been empirically proven) for a binary representation of the environment. Given a sequence of cells, the shortest path through them is a piecewise-linear curve, such that the vertices of the curve, i.e., the starting and ending points snag the convex corners of the sequence. This is similar to the visibility-graph approach in determining a path given a field of polygonal obstacles [Latombe 1991]. Hence the use of corners as a breakpoint to perform optimization of the path up to that point is intuitive because the path changes direction at it cuts through a corner. The uniform cost space, the euclidean geometry and use of A* to search sequences of cells ensures optimality of each polyline, and the sum of polylines gives the optimal path.

For the case of continuously varying cost field, true optimality of the path is not guaranteed. It however gives the best path solution compared with various levels in Method 1. In the continuous case, euclidean geometry cannot be used due to varying costs. Hence in cases where there is a large disparity in costs *locally*, the euclidean distance from center to center generally has a higher cost than the path found using A* and hence this leads to suboptimal solutions. The optimal path is possible if it is not restricted to go just through corners and centers of the cell only. This restriction is however needed in order to keep the search space from exploding. In cases where there is a small disparity in costs locally, the optimal polyline is usually determined. As a result paths passing through only centers and corners of cells need not necessarily give the optimal path. Fig.7 shows the paths obtained by Methods 1 and 2. Although the cost of the path obtained by Method 2 is lower, it is not necessarily optimal.

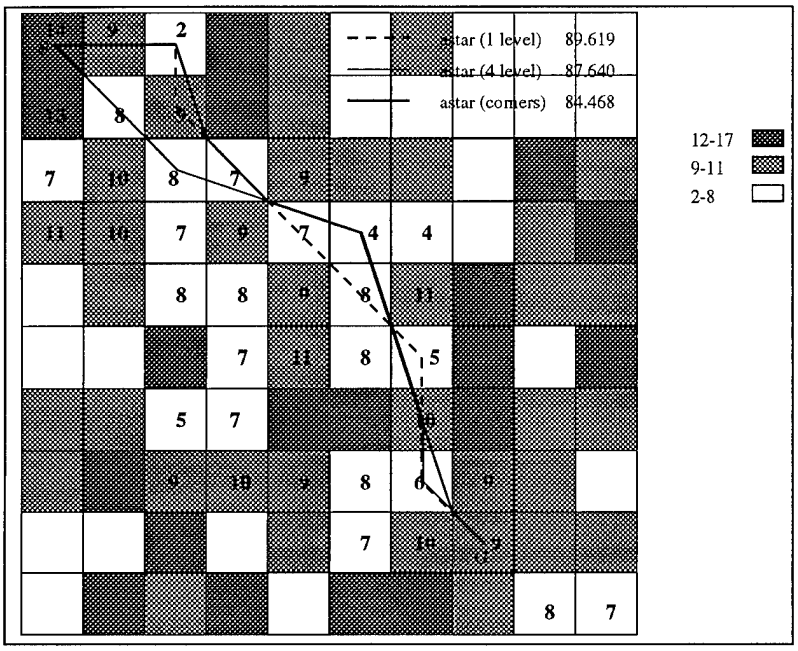


Figure 7: Paths for continuously varying terrain

5 Results

In order to evaluate the two methods, test cases were designed for two types of terrain - binary and continuously varying terrain. 200 cases were run, 100 of grid size 100x100 and the remaining of size 200x200. The total number of obstacles varied from 30% to about 75%. The cost of the path and computing time were recorded.

Computational time-Continuous case

Fig.8 shows the plot of Method 1 (levels 1-4) and Method 2 vs average computational time. It is seen that there is almost a linear increase in time as the number of levels in Method 1 increases. A typical case of 100x100 grid of continuously varying terrain with many obstacles takes 1 second for a Level 1 search. A Level 4 search takes nearly 4 times as much. It is also noted that the computational time for Method 2 is reasonable, taking as much time as a Level 2 search, i.e., it takes roughly 1.7 seconds on average for a 100x100 grid.

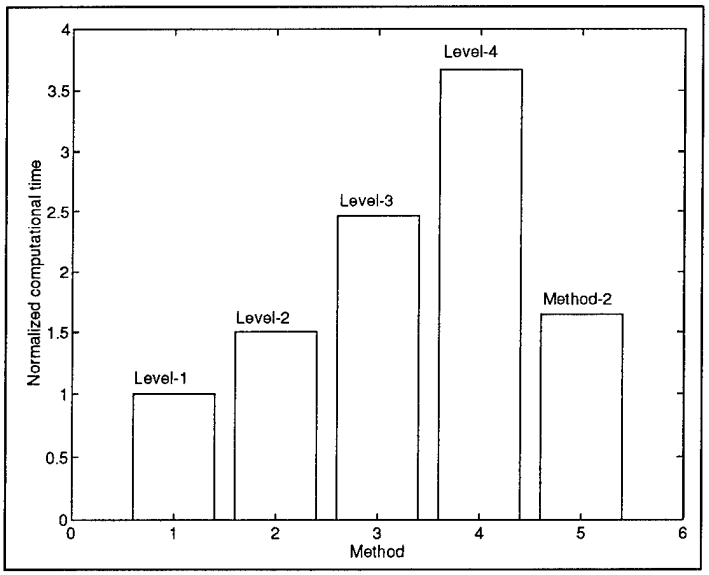


Figure 8: Computational time for Method 1 (up to 4 levels) and Method 2- Continuous case

Computational time-binary case

Fig.9 shows the plot of Method 1 (levels 1-4) and Method 2 vs average computational time for the binary case. Again, there is almost a linear increase in time as the number of levels in Method 1 increases. A typical case of 100x100 grid of binary terrain takes 0.54 seconds for a Level 1 search. A Level 4 search takes nearly 4 times as much. Method 2 takes roughly 1 second on average for a 100x100 grid.

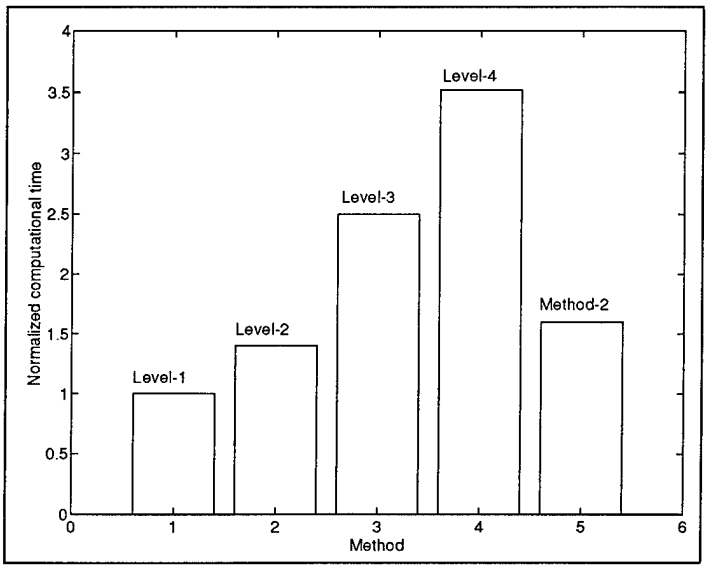


Figure 9: Computational time for Method 1 (up to 4 levels) and Method 2-Binary case

Computational time-Dominance criterion

The advantage of the dominance criterion is illustrated in Fig.10, which shows the difference in times with and without the dominance criterion of the two methods. It is seen that the time saved using this criterion increases significantly as the number of levels used increases. This is because using this criterion reduces the cost computation when a cell is revisited.

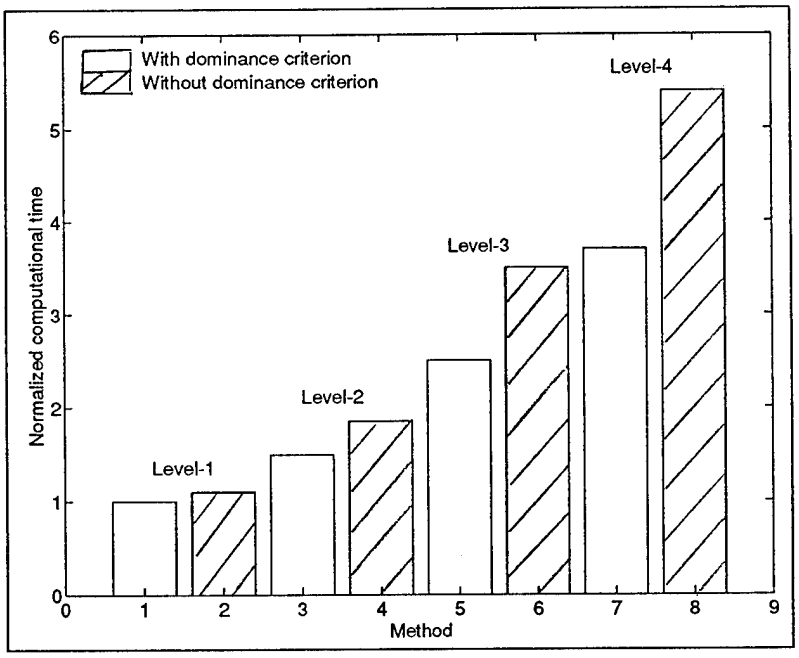


Figure 10: Computational times for Method 1 (up to 4 levels) with and without Dominance Criterion

Accuracy-Continuous case

Fig.11 shows the plot of Method 1 (levels 1-4) and Method 2 vs average normalized cost (it was normalized to lie between 0 and 1, with 0 being the lowest cost and 1 being the highest). It is seen that there is significant (about 40%) cost savings from Level 1 to Level 2 and from Level 2 to Level 3. The difference between levels 3 and 4 however is quite insignificant. Hence the extra effort in terms of computational time is not worth it in the latter case. The cost savings between Method 1 and Method 2 is also shown. Since timewise, Method 2 is comparable to a Level 2 search, the cost savings is nearly 70% of Method 1. Method 2 always performs better.

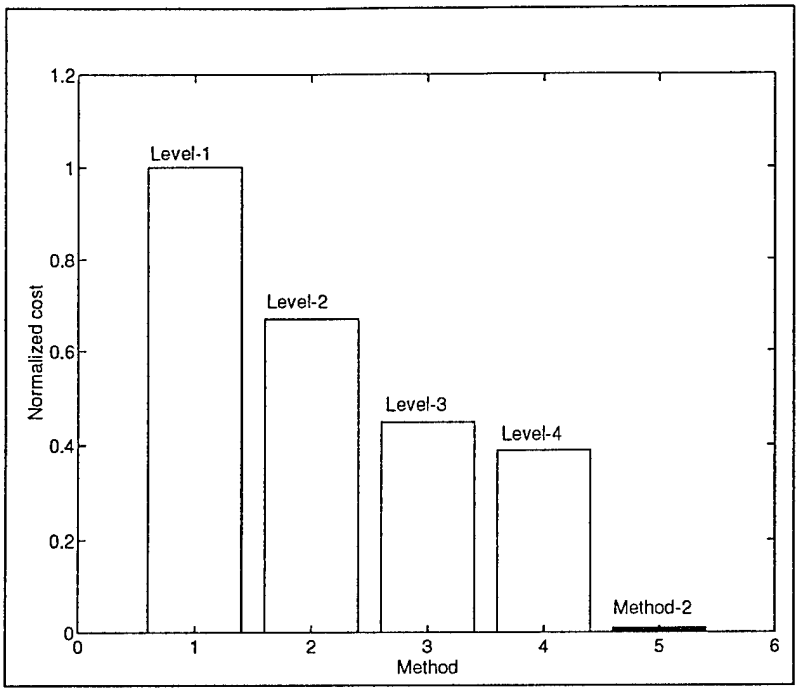


Figure 11: Cost Accuracy for Method 1 (up to 4 levels) and Method 2 - Continuous case

Accuracy-Binary case

Fig.12 shows the plot of Method 1 (levels 1-4) and Method 2 vs average normalized cost for binary case. The cost savings from Level 1 to Level 2 is very significant (nearly 70%) and is much more than for the continuous case. The cost savings between Level 4 and Method 2 is not as significant as for the continuous case. This is due to the relatively flat nature of the terrain. Since timewise, Method 2 is comparable to a Level 2 search, the cost savings is nearly 20% of Method 1. As mentioned earlier, Method 2 gives optimal solutions.

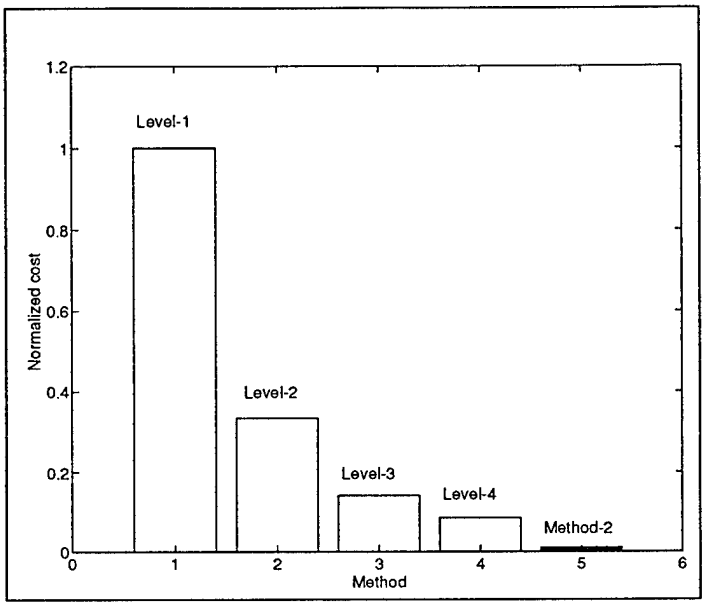


Figure 12: Cost Accuracy for Method 1 (up to 4 levels) and Method 2 - Binary case

Efficiency

Fig.13 shows a plot of the efficiency. It is seen that there is a significant cost saving in increasing the level from 1 to 2 and from 2 to 3. However, the cost reduction in going further from 3 to 4 is marginal. However, the time increase in this case is substantial. It is also noted that Method 2 always performs better than Method 1.

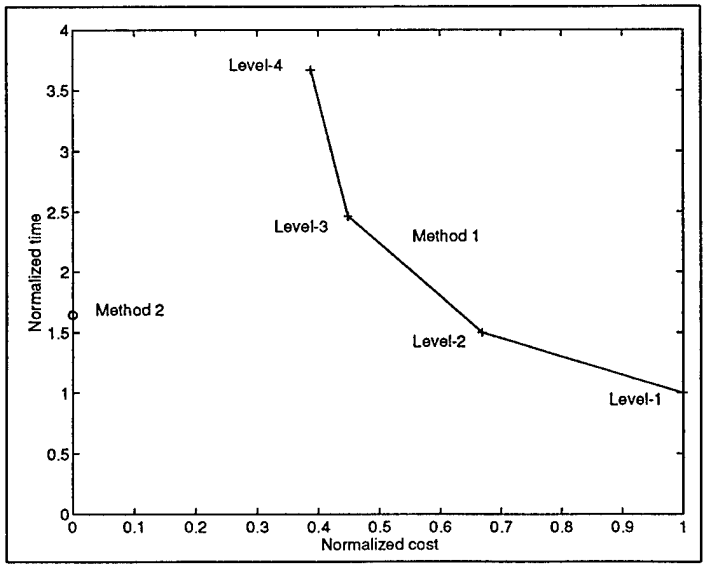


Figure 13: Efficiency frontier

Performance for binary case

Fig.14 and Fig.15 show a simple case which illustrates the advantage and limitations of Method 1. In Fig.14 which is a uniform cost space (10x10 grid) with one obstacle (shaded area), it can be seen that with increasing levels, the path becomes smoother and the cost reduces. In fact, Level 3 finds the truly optimal path. However, as illustrated in Fig.9, the time taken is nearly 3 times as much.

Fig.15 shows a uniform cost space with no obstacles. In this case, it takes up to 5 levels to find the straight line path. Theoretically, it can take N levels to find the optimal path of an NxN space. Hence for N=100, the computational time is tremendous.

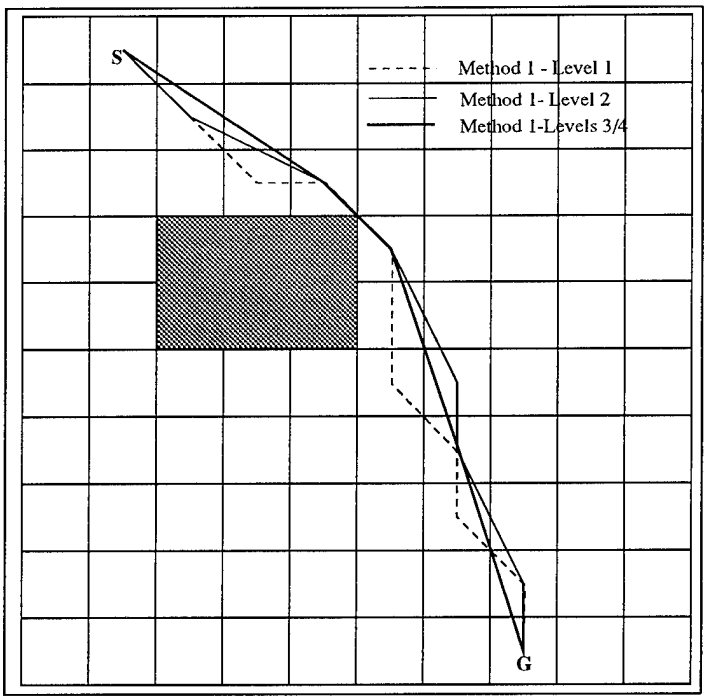


Figure 14: Performance of Method 1 for binary case with obstacle

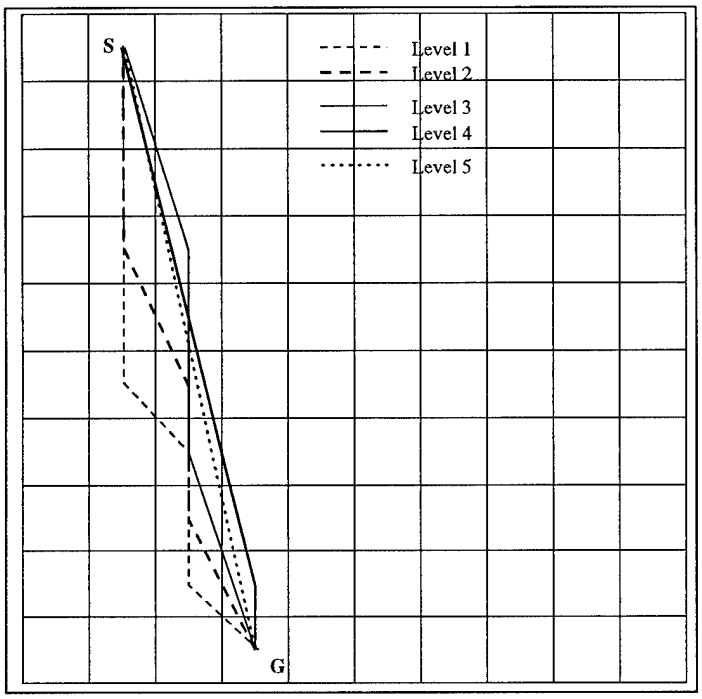


Figure 15: Performance of Method 1 for uniform cost field

Fig.16a-b illustrate the performance of Method 2 vs Method 1 for the binary case .

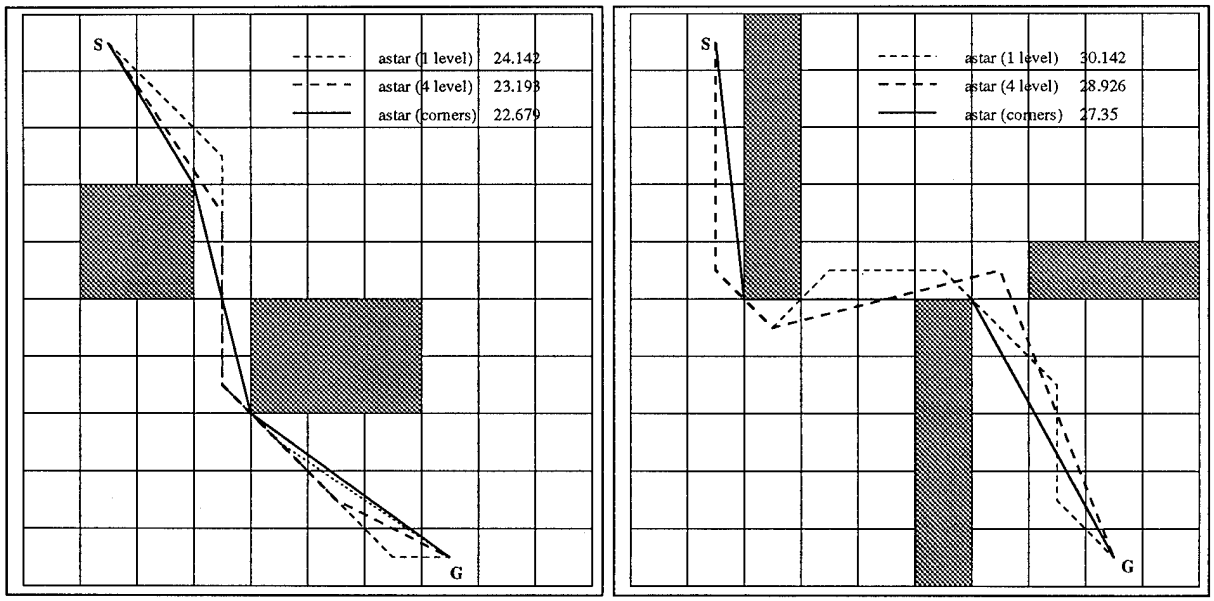


Figure 16: Performance of Method 1 and Method 2 for binary case

6 Conclusions

There are certain situations where the optimality of the path found becomes important. One such case is the minimization of energy usage for long duration missions.

The algorithm developed produces a truly optimal solution for a binary environment and a solution of very high accuracy (as compared to common existing techniques) for the continuously varying space. In order for it to be used in real situations, some of its limitations need to be addressed. Firstly, the robot is considered to be a point. In the binary case, usually the obstacles are grown by a fixed amount to justify this assumption. In the continuously varying case, it is not as straightforward. A method to account for this assumption needs to be developed. Secondly, the algorithm needs to be extended to planning in real-time with uncertainty in the environment.Thirdly, for the algorithm to truly minimize a parameter, like energy, the cost values for each cell have to be realistically calculated. In this case, there are other factors beside locomotion that consume power. Such factors have to be incorporated into the cost of the cells.

References

[1] Barraquand, J. and Latombe, J.C. (1989) "Robot Motion Planning: A Distributed Representation Approach," International Journal of Robotics Research, Vol.10, no.6, 628-49.

[2] Brooks, R.A. (1982) "Symbolic Error Analysis and Robot Planning," International Journal of Robotics Research,Vol.1, no.4, 29-68.

[3] Chatila, R. (1982) "Path Planning and Environment Learning in a Mobile Robot System," Proceedings of the European Conference on Artificial Intelligence, Orsay, France.

[4] Dufay, B. and Latombe, J.C. (1984) "An Approach to Automatic Robot Programming Based on Inductive Learning," International Journal of Robotics Research,Vol.3, no.4, 3-20.

[5] Jarvis, R.A. (1985) "Collision-Free Trajectory Planning Using the Distance Transform," Mechanical Engineering Transactions of the Institute of Engineers, Australia, Vol.ME10, no.3.

[6] Khatib, O. (1986) "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," International Journal of Robotics Research, Vol.5, no.1, 90-98.

[7] Latombe, J.C. (1991) "Robot Motion Planning," Kluwer Academic Publishers, Boston.

[8] Lozano-Perez, T. (1983) "Spatial Planning: A Configuration Space Approach," IEEE Transactions on Computers, Vol.C-32, no.2, 108-120.

[9] Nilsson, N.J. (1980) "Principles of Artificial Intelligence," Tioga Publishing Company.

[10] Stentz, A. (1994) "Optimal and Efficient Path Planning for Partially-Known Environments," Proceedings of 1994 IEEE International Conference on Robotics and Automation, San Deigo.

[11] Taylor, R.H. (1976) "Synthesis of Manipulator Control Programs from Task-Level Specifications," Ph.D. Dissertation, Department of Computer Science, Stanford University.

[12] Zelinsky, A. (1992) "A Mobile Robot Exploration Algorithm," IEEE Transactions of Robotics and Automation, Vol.8, no.6, 707-17.